Quad Core Processor Overview

Brian Waldecker, Ph.D.
Senior Member of Technical Staff
AMD, Austin
2/26/2007

Talk Outline

AMD
Smarter Choice

1. Multi-Core Processor Architecture
2. FPU enhancements
3. Core IPC enhancements
4. Cache Hierarchy and Structure
5. TLBs and large pages
6. NB enhancements
7. Prefetching
8. Some Recommended Programming Practices

Introducing “Barcelona”…

Native quad-core upgrade for 2007

AMD Smarter Choice

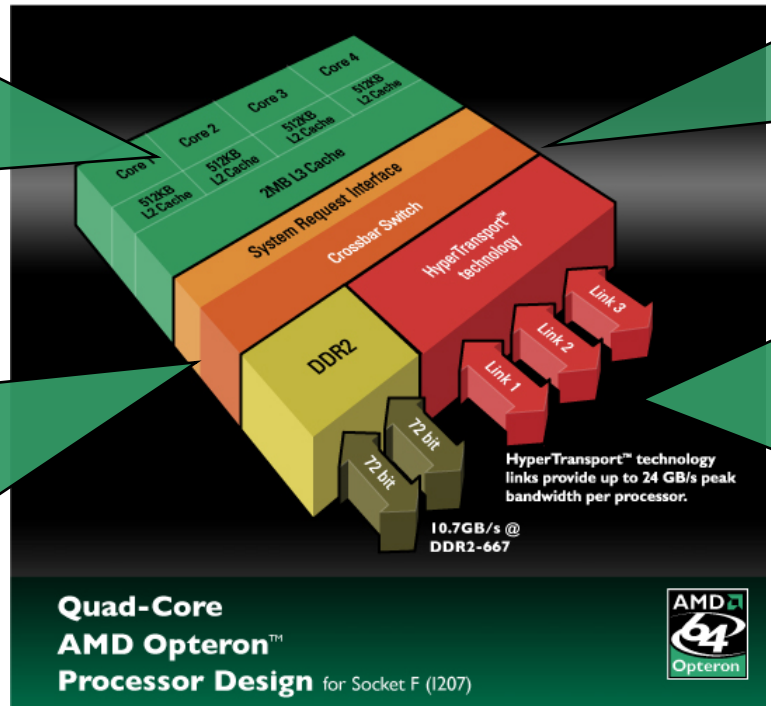
Native Quad-Core Processor

To increase performance-per-watt efficiencies **using the same Thermal Design Power.**

Advanced Process Technology

65nm Silicon-on Insulator Process

Fast transistors with low power leakage to **reduce power and heat.**



Platform Compatibility

Socket and thermal compatible with “Socket F”.

Direct Connect Architecture

- Integrated memory controller designed for **reduced memory latency and increased performance**
  - Memory directly connected
- Provides **fast CPU-to-CPU communication**
  - CPUs directly connected

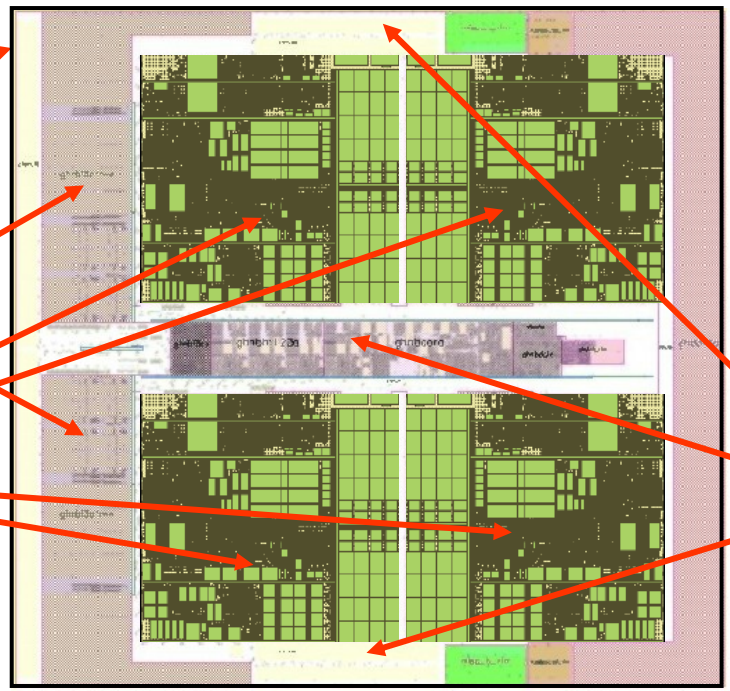
AMD's Next Generation Processor Technology

Optimized for 65nm SOI and beyond

Native quad core die

Expandable shared L3 cache

IPC enhanced CPU cores



Enhanced Direct Connect Architecture and Northbridge

- 32B instruction fetch
- Improved branch prediction
- Out-of-order load execution
- Up to 4 DP FLOPS/cycle
- Dual 128-bit SSE dataflow
- Dual 128-bit loads per cycle
- Instruction Extensions
- Power Management Extensions

- Enhanced HyperTransport™ links
- Enhanced crossbar
- DDR2 with migration path to DDR3
- Enhanced power management and RAS

Comprehensive Upgrades for SSE128

128bit FPU

Parameter	Current Processor	"Barcelona"
SSE Exec Width	64	128 + SSE MOVs
Instruction Fetch Bandwidth	16 bytes/cycle	32 bytes/cycle + unaligned Ld-Ops
Data Cache Bandwidth	2 x 64bit loads/cycle	2 x 128bit loads/cycle
L2/NB Bandwidth	64 bits/cycle	128 bits/cycle
FP Scheduler Depth	36 Dedicated x 64-bit ops	36 Dedicated x 128-bit ops

Can perform SSE MOVs in the FP "store" pipe

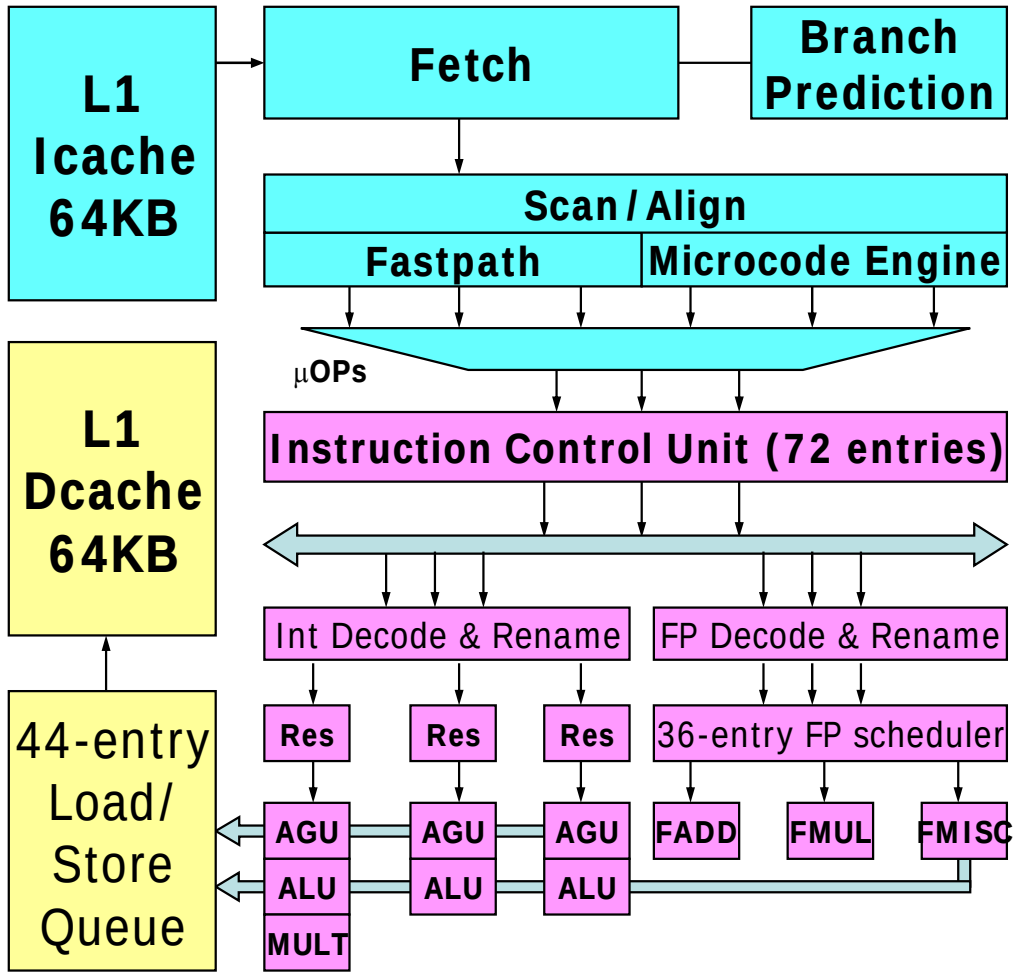
- Execute two generic SSE ops + SSE MOV each cycle (+ two 128-bit SSE loads)

SSE Unaligned Load-Execute mode

- Remove alignment requirements for SSE ld-op instructions
- Eliminate awkward pairs of separate load and compute instructions
- *To improve instruction packing and decoding efficiency*

Core IPC improvements

- Improve Branch Prediction.
- TLB enhancements.
- More out of order Ld/St capability.
- New Instructions
  - POPCNT / LZCNT
  - EXTRQ / INSERTQ
  - MOVNTSD / MOVNTSS
- Fastpath support for FP to Integer data movement.



Cache Hierarchy

AMD Smarter Choice

Dedicated L1 cache

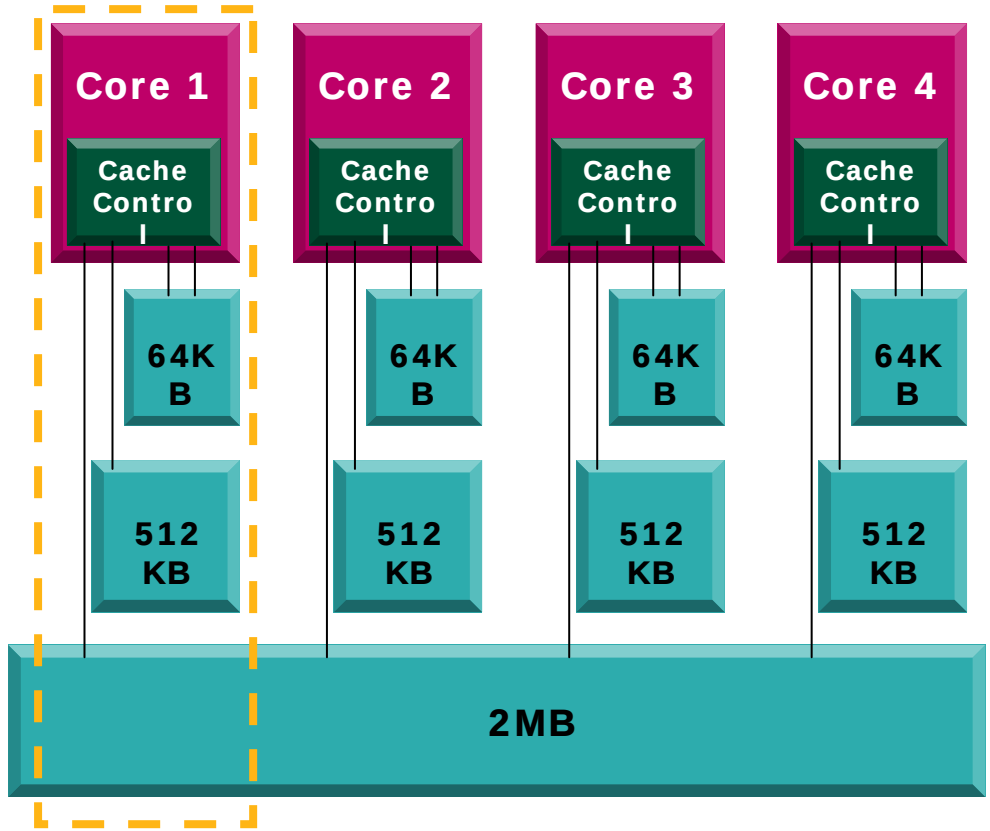
- 2 way associativity.
- 8 banks.
- 2 128bit loads per cycle.

Dedicated L2 cache

- 16 way associativity.

Shared L3 cache

- fills from L3 leave likely shared lines in L3.
- sharing aware replacement policy.



TLB Enhancements

AMD Smarter Choice

- Support for 1GB pagesize (4k, 2M, 1G)
- 48 bit physical addresses = 256TB (increase from previous 40bits)
- Data TLB
  - L1 Data TLB
    - 48 entries, fully associative
    - all 48 entries support any pagesize
  - L2 TLB
    - 512 4k entries, or
    - 128 2M entries
- Instruction TLB
  - L1 Instruction TLB
    - fully associative
    - support for 4k or 2M pagesizes
  - L2 Instruction TLB

Memory Controller Enhancements

- Independent (unganged mode) DRAM controllers
  - allow more concurrent reads of needed data.
  - reduce bank conflicts.
  - longer burst length = better command efficiency.
- Optimized DRAM paging
  - adaptive closing of DRAM pages = more page hits.
- Re-architect Northbridge for higher BW
  - resize buffers, better command scheduling, DDR2 and beyond.
- Write bursting
  - reduce read-write turnarounds.

Data Prefetch

AMD Smarter Choice

- **Hardware prefetching**
  - DRAM prefetcher
    - tracks positive, negative, non-unit strides.
    - dedicated buffer (in NB) to hold prefetched data.
    - Aggressively use idle DRAM cycles.
  - Core prefetchers
    - Does hardware prefetching into L1 Dcache.
- **Software prefetching instructions**
  - MOV (prefetch via load / store)
  - prefetcht0, prefetcht1, prefetcht2 (currently all treated the same)
  - prefetchw = prefetch with intent to modify
  - prefetchnta = prefetch non-temporal (favor for replacement)

Programming Hints - 1

AMD Smarter Choice

- Use xxxpd instructions instead of xxxhpd or xxxsd (e.g. movsd preferred to movlpd, mulpd preferred to mulsd, etc.)

- Beware of false sharing due to independent data residing in the same cacheline

Programming Hints - 2

AMD Smarter Choice

•L1D cache *bank conflicts* will reduce loads hitting in L1 from 2 per cycle to 1.

- L1D cache is implemented as 8 banks. (Software optimization guide gives more detail).
- On K8, if two addresses differ in bits 14:6 but are the same in bits 5:3, a L1D bank conflict is possible (but not definite).
- On Barcelona, bits 5:3 become bits 6:4 (misaligned boundary now 16 bytes)
- Conflict never occurs for walking a single array.
- Conflict is *possible* for certain interleaved access patterns to multiple arrays.

• On Barcelona, ~50% fewer misaligned accesses and 50% improvement to misaligned bandwidth.

Programming Hints - 3

AMD Smarter Choice

- Generally good to prefetch 6 to 8 cachelines ahead

- Try to have 100 cycles of computation in loop body between successive prefetches

- Avoid issuing multiple software prefetches to the same cacheline

- Unroll loops enough times so each iteration works on 1 or more cachelines of data.

note: neither hw or sw prefetches will be allowed to generate page faults, but a TLB miss on a prefetch can initiate a TLB fill.

Programming Hints - 4

AMD Smarter Choice

Which Prefetch to use ?

Data	Less than ½ L1 size	Less than ½ L2 size or of unknown size		Greater than ½ L2 size
		Reused	Not Reused	
Read only	`prefetch or prefetchnta`	`prefetch`	`prefetchnta`	`prefetchnta`
Sequential read only	`hwprefetcher + prefetch`	`hwprefetcher + prefetch`	`prefetchnta`	`prefetchnta`
Read-write	`prefetchw`	`prefetchw`	`prefetchnta`	`prefetchnta`
Sequential read-write	`prefetchw`	`prefetchw`	`prefetchnta`	`prefetchnta`
Write only	`prefetchw`	`prefetchw`	`movnt`	`movnt`
Sequential write only	`hwprefetcher + prefetchw`	`hwprefetcher + prefetchw`	`movnt`	`movnt`

Summary

Attention paid throughout to improving:

- Core IPC.
- Caches and TLB.
- FPU performance.
- Memory performance.
- Compatibility, Usability, and Programmability.

References

- Opteron Processor Families Technical Documentation
  - http://www.amd.com/us-en/Processors/TechnicalResources/0,,30_182_739_9003,00.html
- *Bios and Kernel Developers Guide* (BKDG)
  - RevF cpus are denoted as “Family 0Fh”.
  - Quadcore BKDG coming.
  - Some portions are useful to others than just Bios and Kernel developers.
- *Software Optimization Guide for AMD64 Processors*
  - Current version applicable through RevF processors.
  - Quadcore version coming.

Acknowledgments

Ben Sander, AMD, Principle Member of Technical Staff

Michael Clark, AMD, AMD Fellow

Mike Goddard, AMD, Director, AMD Performance Labs

Bill Brantley, Ph.D., AMD, Senior Member of Technical Staff

Trademark Attribution